**CHX Security**



**Penetration Test Report of Findings**

**Inlanefreight Ltd.**

Thursday, February 29, 2024

Version 1.0

**Table of Contents**

[Statement of Confidentiality 4](#_Toc161346057)

[Engagement Contacts 5](#_Toc161346058)

[Executive Summary 6](#_Toc161346059)

[Approach 6](#_Toc161346060)

[Scope 7](#_Toc161346061)

[Assessment Overview and Recommendations 7](#_Toc161346062)

[External Network Penetration Test Assessment Summary 9](#_Toc161346063)

[Summary of Findings 9](#_Toc161346064)

[Internal Network Penetration Test Assessment Summary 10](#_Toc161346065)

[Summary of Findings 10](#_Toc161346066)

[External to Internal Network Compromise Walkthrough 11](#_Toc161346067)

[Detailed Walkthrough 12](#_Toc161346068)

[Detailed reproduction steps for this attack chain are as follows: 14](#_Toc161346069)

[Remediation Summary -External Network 49](#_Toc161346070)

[Short Term 49](#_Toc161346071)

[Medium Term 49](#_Toc161346072)

[Long Term 50](#_Toc161346073)

[Remediation Summary -Internal Network 51](#_Toc161346074)

[Short Term 51](#_Toc161346075)

[Medium Term 51](#_Toc161346076)

[Long Term 51](#_Toc161346077)

[Technical Findings Details 52](#_Toc161346078)

[1. Insecure Mount - High 52](#_Toc161346079)

[2. Weak Kerberos Authentication (“Kerberoasting”) - High 54](#_Toc161346080)

[3. Critical patches missing - High 56](#_Toc161346081)

[4. Weak Active Directory Passwords - High 57](#_Toc161346082)

[5. Insecure File Shares - Medium 58](#_Toc161346083)

[6. Enhance Security Monitoring Capabilities - Info 59](#_Toc161346084)

[Appendices 60](#_Toc161346085)

[Appendix A – Finding Severities 60](#_Toc161346086)

[Appendix B – Exploited Hosts 61](#_Toc161346087)

[Appendix C – Compromised Users 62](#_Toc161346088)

[Appendix D – Changes/Host Cleanup 63](#_Toc161346089)

[Appendix E – INLANEFREIGHT.LOCAL Domain Password Review 64](#_Toc161346090)

[Password Statistics 64](#_Toc161346091)

[Most Commonly Used Passwords 64](#_Toc161346092)

[Password Length Breakdown 64](#_Toc161346093)

## Statement of Confidentiality

The contents of this document have been developed by CHX Security. CHX Security considers the contents of this document to be proprietary and business confidential information. This information is to be used only in the performance of its intended use. This document may not be released to another vendor, business partner or contractor without prior written consent from CHX Security. Additionally, no portion of this document may be communicated, reproduced, copied or distributed without the prior consent of CHX Security.

The contents of this document do not constitute legal advice. CHX Security’s offer of services that relate to compliance, litigation or other legal interests are not intended as legal counsel and should not be taken as such. The assessment detailed herein is against a fictional company for training and examination purposes, and the vulnerabilities in no way affect CHX Security external or internal infrastructure.

## Engagement Contacts

|  |  |  |
| --- | --- | --- |
| Inlanefreight Contacts | | |
| **Primary Contact** | **Title** | **Primary Contact Email** |
| Rachel Williams | Chief Executive Officer | rachel@inlanefreight.local |
| **Secondary Contact** | **Title** | **Secondary Contact Email** |
| William Ley | Chief Technical Officer | wley@inlanefreight.local |

|  |  |  |
| --- | --- | --- |
| Assessor Contact | | |
| **Assessor Name** | **Title** | **Assessor Contact Email** |
| CHX Security | Security Consultant | someone@htbacademy.local |

## Executive Summary

Inlanefreight Ltd. (“Inlanefreight” herein) contracted CHX Security to perform a Penetration Test of Inlanefreight’s externally and internally facing network to identify security weaknesses, determine the impact to Inlanefreight, document all findings in a clear and repeatable manner, and provide remediation recommendations.

### Approach

CHX Security performed testing under a “black box” approach Thursday, February 29, 2024 without credentials or any advance knowledge of Inlanefreight’s externally or internally facing environment with the goal of identifying unknown weaknesses. Testing was performed from a non-evasive standpoint with the goal of uncovering as many misconfigurations and vulnerabilities as possible. Testing was performed remotely via a host that was provisioned specifically for this assessment. Each weakness identified was documented and manually investigated to determine exploitation possibilities and escalation potential. CHX Security sought to demonstrate the full impact of every vulnerability, up to and including internal domain compromise. If CHX Security was able to gain a foothold in the internal network, Inlanefreight allowed for further testing including lateral movement and horizontal/vertical privilege escalation to demonstrate the impact of an internal network compromise.

## Scope

|  |  |
| --- | --- |
| External Testing | Internal Testing |
| **10.129.X.X (“External” facing target host)** | **172.16.8.0/23** |
| **\*.inlangefreight.local (all subdomains)** | **172.16.9.0/23** |
|  | **INLANFREIGHT.LOCAL (Active Directory domain)** |

*Table 1: Scope Details*

### Assessment Overview and Recommendations

During the external penetration test against the Inlanefrieght, CHX Security identified ten (10) high-risk findings one (1) low-risk, and one (1) medium-risk finding that threaten the confidentiality, integrity and availability of Inlanefreight’s information systems. One of these findings lead to the compromise of Inlanefreight’s internal network. Once CHX Security gained access to Inlanefreight’s internal network they were able to identify one (1) low-risk, one (1) medium-risk, and four (4) high-risk findings in the internal network.

For the external network the tester identified misconfigurations, multiple services that had high-risk vulnerabilities, as well an exposed services that appear to not be in use anymore.

One of the webservices exposed had a portion exposed that allowed the tester to move inside Inlanefreight’s internal network.

Once the tester gained access to the internal network the tester found that overall, the Inlanefreight’s patch and vulnerability management was well maintained but that there were a couple critical patches that would have prevented further compromise of the network. Most of the flaws were related to misconfiguration and lack of hardening, with most falling under the categories of weak authentication and weak authorization.

One finding was a weak configuration involving service accounts that allows any authenticated user to steal a component of the authentication process that can often be guessed offline (via password “cracking”) to reveal the human-readable form of the account’s password. These types of service accounts typically have more privileges than a standard user, so obtaining one of their passwords in clear text could result in lateral movement or privilege escalation and eventually in complete network compromise. Fortunately, this issue can be corrected without the need for third-party tools. Microsoft’s Active Directory contains settings that can be used to minimize the risk of these resources being abused for the benefit of malicious users.

A webserver was found that the tester was able to find the credentials for on a share that they mounted once they were inside the internal network. The webserver was found to have a SQL console that the tester used to gain a command shell on the server which led to the complete compromise of the server that was hosting the webservice.

The tester also found shared folders with excessive permissions, meaning that all users in the internal network can access a considerable amount of data. While sharing files internally between departments and users is important to day-to-day business operations, wide open permissions on file shares may result in unintentional disclosure of confidential information. Even if a file share does not contain any sensitive information today, someone may unwittingly put such data there thinking it is protected when it isn’t. This configuration should be changed to ensure that users can access only what is necessary to perform their day-to-day duties.

Finally, the tester noticed that testing activities seemed to go mostly unnoticed, which may represent an opportunity to improve visibility into the internal network and indicates that a real-world attacker might remain undetected if internal access is achieved. Inlanefreight should create a remediation plan based on the [Remediation Summary](#_Remediation_Summary_-Internal) section of this report, addressing all high findings as soon as possible according to the needs of the business. Inlanefreight should also consider performing periodic vulnerability assessments if they are not already being performed. Once the issues identified in this report have been addressed, a more collaborative, in-depth Active Directory security assessment may help identify additional opportunities to harden the Active Directory environment, making it more difficult for attackers to move around the network and increasing the likelihood that Inlanefreight will be able to detect and respond to suspicious activity.

## External Network Penetration Test Assessment Summary

CHX Security began all testing activities from the perspective of an unauthenticated user on the external network or from the internet. Inlanefreight provided the tester with an IP address and domain name but did not provide additional information.

### Summary of Findings

During testing, CHX Security uncovered a total of ten (10) high-risk, one (1) medium-risk and one (1) low-risk findings that pose a material risk to Inlanefreight’s information systems. Two subdomains appear that they should not be publicly exposed, and one appears inactive. The below table provides a summary of the findings by severity level.

|  |  |  |  |
| --- | --- | --- | --- |
| Finding Severity | | | |
| **High** | **Medium** | **Low** | **Total** |
| **5** | **1** | **1** | **7** |

*Table 2: Severity Summary*

Below is a high-level overview of each finding identified during testing. These findings are covered in depth in the [Technical Findings Details](#_Technical_Findings_Details) section of this report.

|  |  |  |  |
| --- | --- | --- | --- |
| Finding # | Severity Level | Subdomain | Finding Name |
| 1. | **High** | careers.inlanefreight.local | Insecure Direct Object Reference (IDOR) |
| 2. | **High** | dev.inlanefreight.local | Unrestricted File Upload |
| 3. | **High** | ir.inlanefreight.local | Local File Inclusion (LFI) |
| 4. | **High** | ir.inlanefreight.local | Weak WordPress Admin Credentials |
| 5. | **High** | status.inlanefreight.local | SQL Injection (should not be exposed) |
| 6. | **High** | support.inlanefreight.local | Cross-Site Scripting (XSS) |
| 7. | **High** | tracking.inlanefreight.local | SSRF to Local File Read |
| 8. | **High** | gitlab.inlanefreight.local | Misconfigured GitLab Instance |
| 9. | **High** | shopdev2.inlanefreight.local | XML External Entity (XXE) Injection |
| 10. | **High** | monitoring.inlanefreight.local | Command Injection – to internal network compromise |
| 11. | **Medium** | dev.inlanefreight.local | HTTP Verb Tampering |
| 12. | **Low** | blog.inlanefreight.local | Drupal system that was not up to date and appears not to be in use. |

*Table 3: Finding List*

## Internal Network Penetration Test Assessment Summary

CHX Security gained access to the internal system through a command injection vulnerability and began testing activities from the perspective of an authenticated user (access gained from the external facing systems) on the network. Inlanefreight provided the tester with network ranges but did not provide additional information such as operating system or configuration information.

### Summary of Findings

During testing, CHX Security uncovered a total of seven (7) findings that pose a material risk to Inlanefreight’s information systems. CHX Security also identified one informational finding that, if addressed, could further strengthen Inlanefreight’s overall security posture. Informational findings are observations for areas of improvement by the organization and do not represent security vulnerabilities on their own. The below table provides a summary of the findings by severity level.

|  |  |  |  |
| --- | --- | --- | --- |
| Finding Severity | | | |
| **High** | **Medium** | **Low** | **Total** |
| **5** | **1** | **1** | **7** |

*Table 4: Severity Summary*

Below is a high-level overview of each finding identified during testing. These findings are covered in depth in the [Technical Findings Details](#_Technical_Findings_Details) section of this report.

|  |  |  |
| --- | --- | --- |
| Finding # | Severity Level | Finding Name |
| 1. | **High** | Insecure Mount that had plaintext password on it |
| 2. | **High** | Weak Kerberos Authentication (“Kerberoasting”) |
| 3. | **High** | Critical patches missing |
| 4. | **High** | Weak Active Directory Passwords |
| 6. | **Medium** | Insecure File Shares |
| 7. | **Info** | Enhance Security Monitoring Capabilities |

*Table 5: Finding List*

## External to Internal Network Compromise Walkthrough

During the course of the assessment CHX Security was able to find multiple subdomains that had high-risk findings including a Command Injection that led to the compromise of Ilanefreight’s local network systems and a full domain compromise. The steps below demonstrate the steps taken from initial access to compromise and does not include all vulnerabilities and misconfigurations discovered during testing. Any issues not used as part of the path to compromise are listed as separate, standalone issues in the [Technical Findings Details](#_Technical_Findings_Details) section, ranked by severity level. The intent of this attack chain is to demonstrate to Inlanefreight the impact of each vulnerability shown in this report and how they fit together to demonstrate the overall risk to the client environment and help to prioritize remediation efforts (i.e., patching two flaws quickly could break up the attack chain while the company works to remediate all issues reported). While other findings shown in this report could be leveraged to gain a similar level of access, this attack chain shows the initial path of least resistance taken by the tester to achieve domain compromise.

## Detailed Walkthrough

CHX Security performed the following to fully compromise the INLANEFREIGHT.LOCAL domain.

1. The tester used a command injection vulnerability found in the monitoring.inlanefreight.local domain to gain access as the user webdev on the host DMZ01. This user was a member of the (adm) group which has rights to read all the log files.
2. The tester found another user, srvadm, and their password shown in plaintext in a log.
3. The tester was able to use these credentials to gain persistence on the DMZ01 host by using the credentials with SSH.
4. Once connected via SSH the tester found that the user srvadm was allowed to run /usr/bin/openssl as the sudo user on the DMZ01 host.
5. Using these rights the tester was able to read the SSH of the root user on DMZ01 and fully compromise the DMZ01 host.
6. Using this access and Ligolo-NG the tester was able to setup the DMZ01 host as a pivot point to dig deeper into the Inlanefreight network.
7. Once the pivot was set up the tester was able to identify three more hosts on the network, during enumeration the hosts DEV01 (IP 172.16.8.20), DC01(IP 172.16.8.3) and MS01(IP 172.16.8.50) were identified. Notably DC01 was a domain controller.
8. Enumerating further the tester found DotNetNuke (DNN) running on the DEV01 host.
9. The tester also found a share that they were able to mount and read, as well as download files from.
10. There was a web.config file on the Shares that contained Administrator login credentials to the DNN webapp running on DEV01.
11. After logging into the DNN webapp the tester found a SQL Console that they were able to exploit to get command shell access on the host DEV01.
12. Using this access the tester found the user had SeImpersonate rights on the host.
13. With this in mind the tester was able to compromise the host with the “Print Nightmare” vulnerability. Using PrintSpoofer64.exe and nc.exe the tester was able to elevate their rights to the NT AUTHORITY\SYSTEM user, fully compromising the DEV01 host.
14. With the newly established rights on DEV01 the tester was able to dump the SAM database and get the Administrator’s hash as well as a plaintext password.
15. Dumping LSA again with CrackMapExec the tester was able to confirm that the plaintext password was a domain credential for the user hporter.
16. Hporter account had ForceChangePassword rights over the ssmalls user.
17. Using PowerView.ps1 the tester was able to change the password on the ssmalls user.
18. Using ssmalls user and going back through the shares found earlier the is a Backup.ps1 script that contains the user backupadm and their password.
19. Using the backupadm credentials the tester could connect to the MS01 Host.
20. Enumerating the MS01 host the tester found an unattend.xml file, with credentials for the user ilfserveradm who seems to not be a domain user.
21. Using ilfserveradm the tester was able to connect with RDP to MS01 and found that there was a service running, sysaxschedscp.exe that was exploitable to gain local administrator rights on the MS01 host.
22. Running lsadump with mimikatz the tester was able to find a password which upon querying the domain they discovered was for the account mssqladm that was being used for autologn.
23. Mssqladm has GenericWrite over the user ttimmons which was used to create a fake SPN and perform a targeted kerberoasting attack.
24. Back on the DEV01 machine the tester user PSCredential object to run commands as the mssqladm user where they used Set-DomainObject to set a fake SPN on the target account (ttimmons).
25. Back on the attack host the tester used GetUserSPN.py to perform a targeted Kerberoasting attack and got the users hash of their password.
26. Using Hashcat on this hash the tester was able to crack ttimmons password.
27. Checking BloodHound the tester found that ttimmons had GenericAll rights over the Server Admins group.
28. The Server Admins group appeared to have the ability to perform DCSync and obtain the NTLM password hash for any users in the domain.
29. Using PSCredential object again the tester added ttimmons to the Server Admins Group.
30. Finally, the tester used Secretsdump to DCSync all NTLM password hashes from the Domain Controller.
31. The tester than ran hashcat against all the password hashes and was able to find a few more passwords of users and to obtain the Administrator’s hash.
32. The tester was able to use the Administrator’s hash to connect to the Domain Controller host DC01.
33. Enumerating DC01 the tester found that the host had dual-homed NICs and found another network.
34. Scanning this network the tester found another host MGMT01 with an IP of 172.16.9.25
35. The tester also found SSH keys in a share that they could now access as the Administrator.
36. Using one of these SSH keys the tester was able to connect to the MGMT01 host after setting up a new pivot point with Ligolo-NG.
37. After connecting to MGMT01 the tester was able to discover that this was a Linux machine that was running an older kernel that was vulnerable to the dirtypipe exploit.
38. The tester was able to exploit this and gain root access to the MGMT01 host fully compromising it and fully compromising the Domain of Inlanefreight.

## Detailed reproduction steps for this attack chain are as follows:

After scanning the domain for subdomains the tester connected to monitoring.inlanefreight.local where they were redirected to a login page.

A black rectangular object with a black border

Description automatically generated

*Figure 1: login screen*

On this page the test used the hydra tool to test for passwords with the username admin and was able to find the credentials to login.

A screenshot of a computer program

Description automatically generated

*Figure 2: Using Hydra*

Using this password the tester logged into the site where they found a monitoring console. The tester was able to bypass the filters in place on this console and gain a reverse shell into the host as the user webdev.

A screen shot of a computer

Description automatically generated

*Figure 3: Using Burpsuite for filter bypass*

A screenshot of a computer program

Description automatically generated

*Figure 4: Using Burpsuite for Filter Bypass*

A computer screen with white text

Description automatically generated

*Figure 5: Running Command on attack host.*

After gaining a reverse shell on the DMZ01 host the tester found they were a member of the adm group and could read logs. After reading some logs they were able to find cleartext credentials for the user srvadm.

A screenshot of a computer program

Description automatically generated

*Figure 6: Finding Cleartext Password in log.*

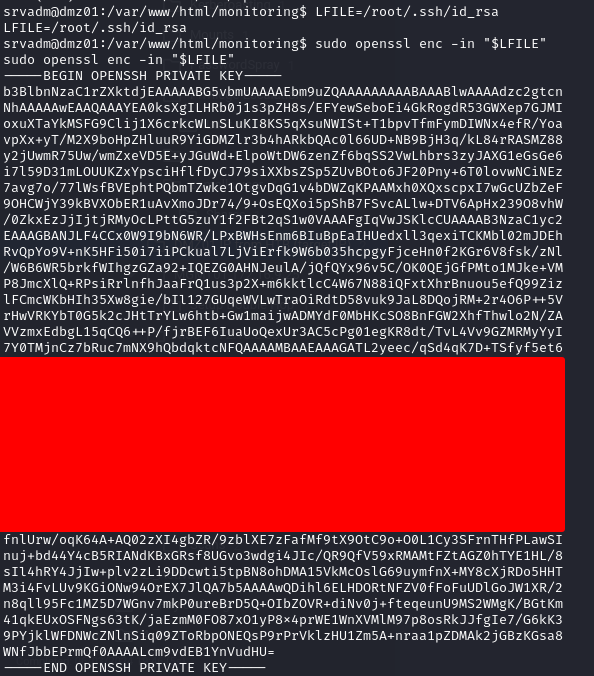
The tester was able to use these credentials to login to the host as the user srvadm and discover that srvadm could run /usr/bin/openssl as the root user.

A screenshot of a computer screen

Description automatically generated

*Figure 7: Logging in as srvadm User on DMZ01*

With this knowledge the tester was able to read the root user’s SSH key to gain access to the host as the root user via SSH.



*Figure 8: Finding SSH Key for root user.*

A screenshot of a computer

Description automatically generated

*Figure 9: Using SSH key to connect to DMZ01 as root user.*

Next the tester set up ligolo-ng to use as a pivot from the attack machine to the 172.16.8.0/24 network.



*Figure 10: Using Ligolo-NG to setup a tunnel.*

A screenshot of a computer

Description automatically generated

*Figure 11: Using Ligolo-NG to set up a tunnel.*

A screenshot of a computer

Description automatically generated

*Figure 12: Using Ligolo-NG to show ifconfig settings.*

Next the tester scanned the subnet for any live hosts.

A screen shot of a computer

Description automatically generated

*Figure 13: Running nmap scan on internal network.*

The tester found three hosts that appeared to be live and did further scanning on them.

**DEV01**

A screen shot of a computer program

Description automatically generated

*Figure 14: Running nmap scan on internal network.*

**MS01**

A screenshot of a computer program

Description automatically generated

*Figure 15: Running nmap scan on internal network.*

**DC01**

A screenshot of a computer

Description automatically generated

*Figure 16: Running nmap scan on internal network.*

The tester found that DotNetNuke (DNN) was running on DEV01 (172.16.8.20)

A screenshot of a computer

Description automatically generated

*Figure 17: DonNetNuke login page.*

There was a login button for DNN

A screenshot of a login box

Description automatically generated

*Figure 18: DonNetNuke login page.*

Looking further into DEV01 the tester found that there was a share that they could mount and access files from. On this share they found a web.config file that had plaintext credentials for user Administrator.

A screenshot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

*Figure 19: Mounting NFS on DEV01*

A screenshot of a computer program

Description automatically generated

*Figure 20: Finding Credentials in web.config file.*

The tester was able to use these credentials to log into the DNN webpage. Once logged in they found a SQL Console that they were able to use to run commands on the DEV01 host.

A screenshot of a computer

Description automatically generated

*Figure 21: DonNetNuke sqlconsole.*

A screenshot of a computer

Description automatically generated

*Figure 22: DonNetNuke sqlconsole.*

A screenshot of a computer

Description automatically generated

*Figure 23: DonNetNuke sqlconsole.*

Using this webshell the tester was able to gain a reverse shell connection on DEV01.

First they needed to add a listener to Ligolo-ng that was pointing to the internal IP on DEV01 of 172.16.8.120

Ligolo-NG command to add listener

**listener\_add --addr 0.0.0.0:1235 --to 127.0.0.1:9001**

A screen shot of a computer

Description automatically generated

*Figure 24: DonNetNuke sqlconsole base64 encoded reverse shell command.*

A screenshot of a computer program

Description automatically generated

*Figure 25: Getting a shell on DEV01.*

Next the tester, after checking the users rights, found that the user has SeImpersonatPrivilege. They were able to use these rights to escalate privileges to the nt authority\system.

Another listener was setup on the Ligolo-ng proxy



*Figure 26: Adding a route to Ligolo-NG*

Files were copied from the attack host to the DEV01 host using the Ligolo-ng proxy

A computer screen shot of a computer

Description automatically generated

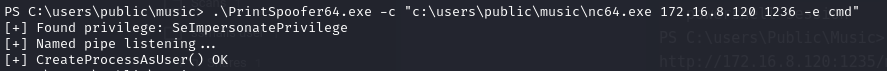
*Figure 27: Moving Files to DEV01*

Another listener was setup with Ligolo-ng

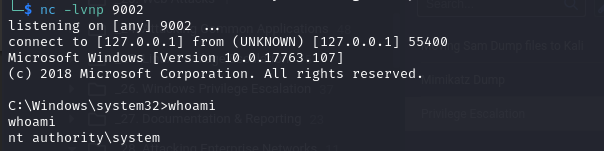


*Figure 28: Adding a listener to Ligolo-NG*

A listener was setup on the attack host and after the command was ran a shell was gained as nt authority\system.



*Figure 29: Exploiting PrintSpoofer64 to get an Admin Shell on DEV01*



*Figure 30: Getting a nt authority\system shell on DEV01.*

From here the tester retrieved the contents of the SAM database and with the local administrator password hash.

A screen shot of a computer

Description automatically generated

*Figure 31: Dumping the SAM files on DEV01.*

The tester then moved this data back to their attack host,

A screenshot of a computer

Description automatically generated

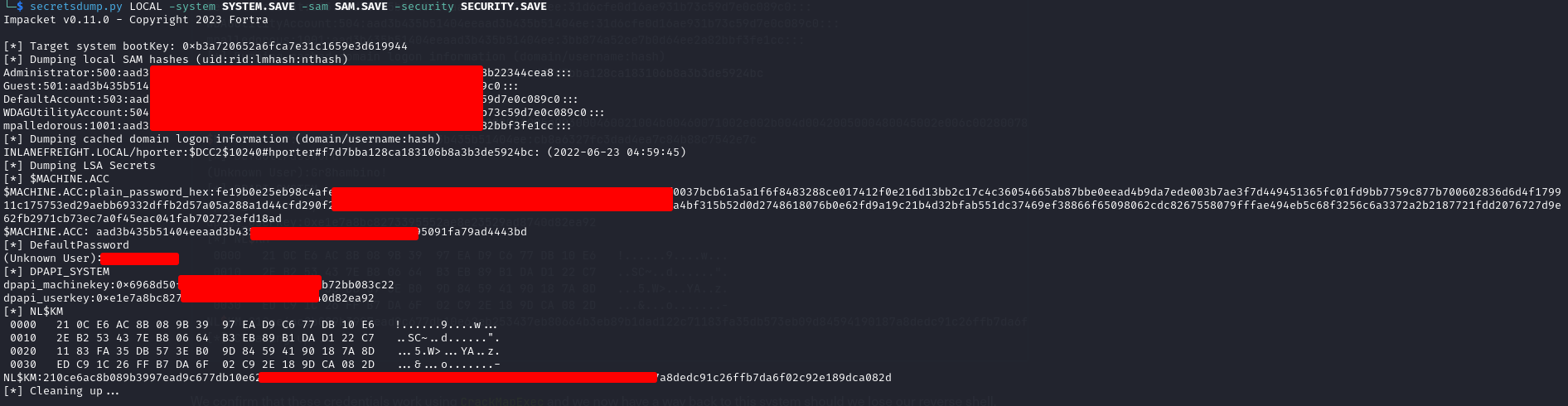
*Figure 32: Adding another listener to Ligolo-NG*

A screenshot of a computer program

Description automatically generated

*Figure 33: Moving the SAM dump back to the attack host.*

Next the tester used secretsdump to dump the SAM database and retrieve a set of credentials from LSA secrets. This also included a plaintext password



*Figure 34: Running secrecsdump on the SAM files and finding a cleartext password.*

The tester confirmed that credentials for Administrator work using crackmapexec



*Figure 35: Confirming credentials for the Administrator user.*

The tester confirmed that the plaintext password was for the user hporter.

A red and black text

Description automatically generated with medium confidence

*Figure 36: Confirming password found was for hporter user.*

The tester confirmed that the user hporter was a domain user with the reverse shell they had.

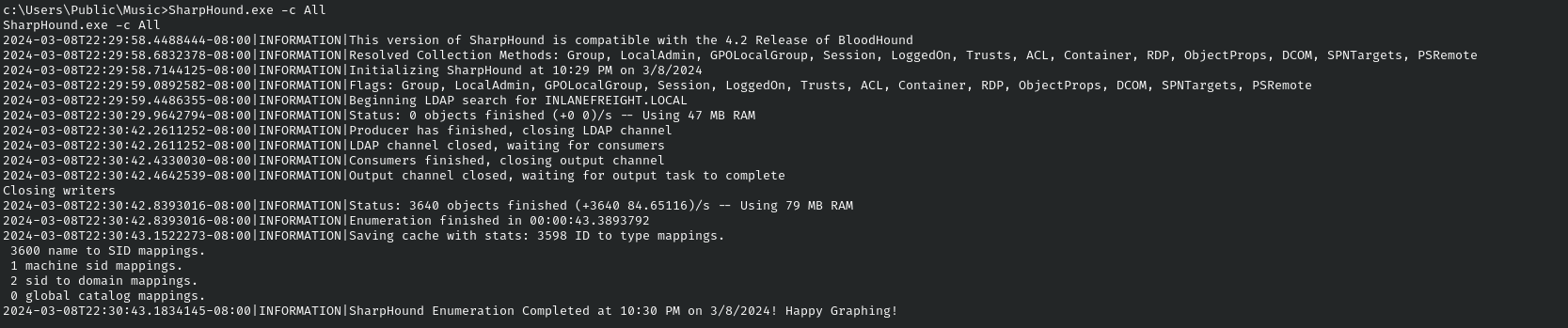
A screenshot of a computer

Description automatically generated

*Figure 37: Confirming hporter user is a domain user.*

The tester then moved SharpHound.exe over and ran it.





*Figure 38: Running SharpHound.exe on the host.*

After moving the BloodHound.zip file over to the attack host the tester uploaded the data into the bloodhound tool and looked at the rights the user hporter had, this showed that hporter had ForceChangePassword rights over the user ssmalls.

A close-up of a black background

Description automatically generated

*Figure 39: Checking that hporter has ForceChangePassword rights over ssmalls user.*

After connecting as the user hporter to DEV01 the tester moved the PowerView tool over to DEV01 to force a password change on the ssmalls user.

A screen shot of a computer

Description automatically generated

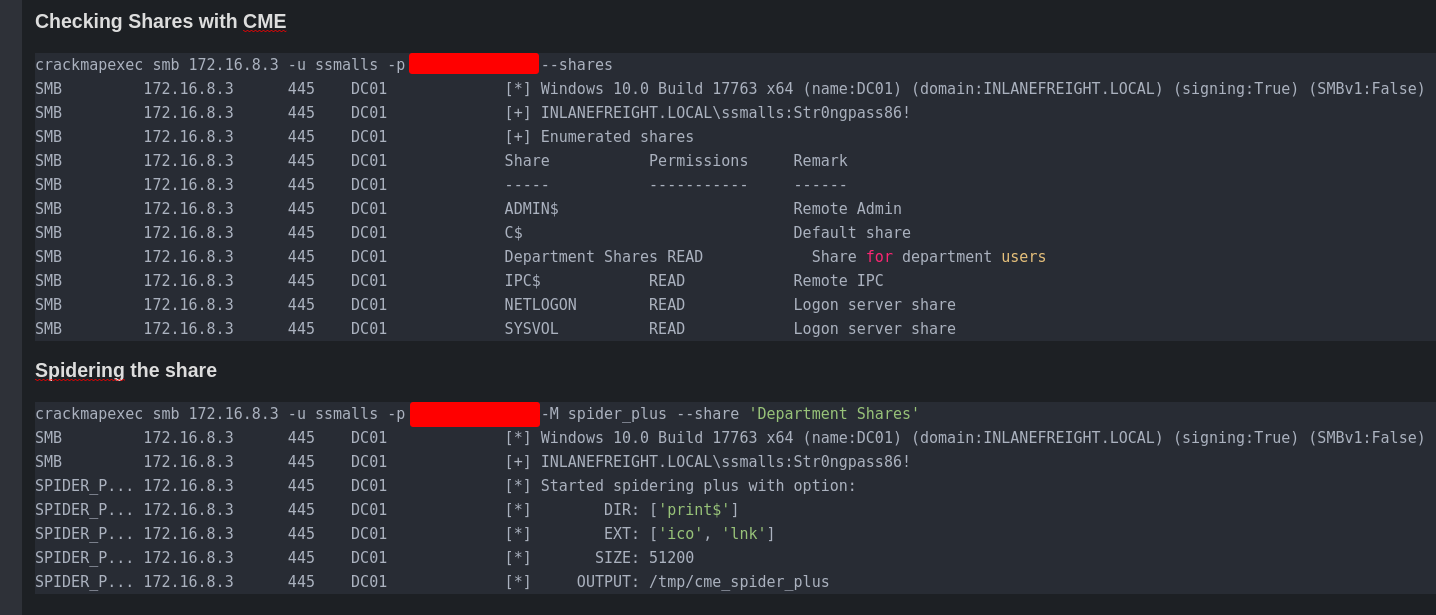
*Figure 40: Connecting to DEV01 with hporter user.*

A computer screen with text on it

Description automatically generated

*Figure 41: Changing password for ssmalls user.*

Using the ssmalls user the tester enumerated shares and found that there was a Backup.ps1 script that had credentials for the user ilfserveradm.



*Figure 42: Looking at shares with ssmalls.*

A screenshot of a computer

Description automatically generated

*Figure 43: Viewing file found on share.*

Connecting to the share with smbclient A screenshot of a computer

Description automatically generated

*Figure 44: Connecting to the share and downloading the script.*

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

*Figure 45: Finding credentials in the script file.*

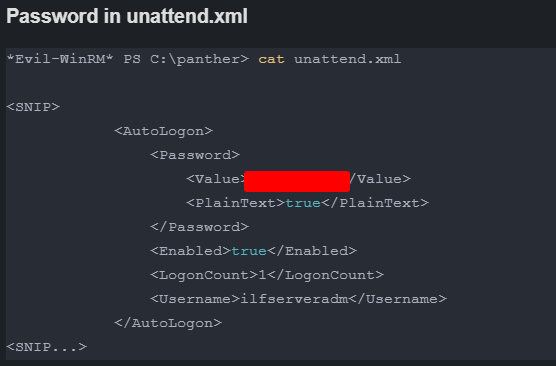
Using these credentials the tester was able to connect to the host MS01 as the user backupadm

A screenshot of a computer

Description automatically generated

*Figure 46: Connecting to MS01 with backupadm and the password found.*

The tester found an xml file on MS01 that had cleartext password for the ilfserveradm in it.



*Figure 47: Finding a password for ilfserveradm user in the unattend.xml file.*

Using these credentials the tester was able to RDP into the host.



*Figure 48: Connecting to the host with ilfserveradm user.*

Running lsadump with mimikatz the tester was able to find a password which upon querying the domain they discovered was for the account mssqladm that was being used for autologn.

After RDPing in and performing additional enumeration, the tester found some non-standard software installed in the C:\\Program Files (x86)\\SysaxAutomation

A screenshot of a computer

Description automatically generated

*Figure 49: Running lsadump with mimikatz.*

A screenshot of a computer

Description automatically generated

*Figure 50: Running lsadump with mimikatz.*

Mssqladm has GenericWrite over the user ttimmons which can be used to create a fake SPN and perform a targeted kerberoasting attack.

A black background with white text

Description automatically generated

*Figure 51: Checking mssqladmi has GenericWrite over user ttimmons.*

Back on the DEV01 machine the tester used PSCredential object to run commands as the mssqladm user where the used Set-DomainObject to set a fake SPN on the target account (ttimmons).

A screenshot of a computer

Description automatically generated

*Figure 52: Setting fake SPN account on ttimmons user.*

Back on the attack host the tester used GetUserSPN.py to perform a targeted Kerberoasting attack and got the users hash of their password.

A screen shot of a computer

Description automatically generated

*Figure 53: Kerberoasting attack on ttimmons.*

Using Hashcat on this hash the tester was able to crack ttimmons password.

A screenshot of a computer

Description automatically generated

*Figure 54: Running hashcat on ttimmons hash and finding their password.*

Checking BloodHound the tester found that ttimmons had GenericAll rights over the Server Admins group.

A black background with white text

Description automatically generated

*Figure 55: Checking that ttimmons has GenericAll rights over Server Admins group.*

The Server Admins group appeared to have the ability to perform DCSync and obtain the NTLM password hash for any users in the domain.

A diagram of a diagram

Description automatically generated

*Figure 56: Checking that server admins group and DCSync to the DC.*

Using PSCredential object the tester added ttimmons to the Server Admins Group.

A screen shot of a computer

Description automatically generated

*Figure 57: Adding ttimmons to the server admins group.*

Finally, the tester used Secretsdump to DCSync all NTLM password hashes from the Domain Controller including the Administrators hash.

A screenshot of a computer

Description automatically generated

*Figure 58: DCSync and dumping the domains hashes.*

The tester than ran hashcat against all the password hashes and was able to find a few more passwords of users.

A screenshot of a computer

Description automatically generated

*Figure 59: Running Hashcat against the domain hashes.*

This was Guest, ttimmons, mmertile and backupjob passwords that were found.

Next the tester was able to use the Administrators hash to connect to the Domain Controller host DC01.

A black screen with white text

Description automatically generated

*Figure 60: Using the Administrators hash in a pass the hash attack to connect to the DC.*

The tester then ran ipconfig and found the Domain Controller was dual-homed

A screenshot of a computer

Description automatically generated

*Figure 61: Checking the DC to see if it is dual-homed.*

Scanning the newly discovered network the tester found another host MGMT01 with an IP of 172.16.9.25

A screen shot of a computer

Description automatically generated

*Figure 62: Scanning the new network found.*

A screenshot of a computer program

Description automatically generated

*Figure 63: Scanning the new network found.*

The tester also found SSH keys in a share that they could now access since they were logged in as the Administrator.

A screen shot of a computer

Description automatically generated

*Figure 64: Enumerating the DC and finding SSH keys.*

Using one of these SSH keys the tester was able to connect to the MGMT01 host after setting up a new pivot point with Ligolo-ng

A screenshot of a computer program

Description automatically generated

*Figure 65: Setting up the double pivot point with Ligolo-NG*

A screen shot of a computer

Description automatically generated

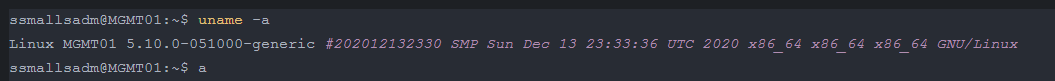
*Figure 66: Starting a tunnel back to the attack host.*

A screenshot of a computer

Description automatically generated

*Figure 67: Connecting to MGMT01 via SSH key found on DC.*

After connecting to MGMT01 the tester was able to discover that this was a linux machine that was running an older kernel that was vulnerable to the dirtypipe exploit.



*Figure 68: Checking Kernel version on the MGMT01 host.*

The tester was able to exploit this and gain root access to the MGMT01 host fully compromising it and fully compromising the Domain of Inlanefreight.

The tester moved the exploit to MGMT01

A black background with white text

Description automatically generated

*Figure 69: Moving the dirypipe exploit over to MGMT01*

Then the tester ran the exploit and was able to gain root (administrator) rights on the machine.

A screenshot of a computer

Description automatically generated

*Figure 70: Running dirtypipe and gaining root privileges on MGM01.*

## Remediation Summary -External Network

As a result of this assessment there are several opportunities for Inlanefreight to strengthen its internal network security. Remediation efforts are prioritized below starting with those that will likely take the least amount of time and effort to complete. Inlanefreight should ensure that all remediation steps and mitigating controls are carefully planned and tested to prevent any service disruptions or loss of data.

### Short Term

* **Finding 1** – Verify the user's permission every time an access attempt is made. Implement this structurally using the recommended approach for your web framework.
* **Finding 2** – Implementing a defense in depth approach is key to make the upload process harder and more locked down to the needs and requirements for the service. Implementing multiple techniques is key and recommended, as no one technique is enough to secure the service.
* **Finding 3** – use verified and secured whitelist files and ignore everything else
* **Finding 4** – Set strong passwords (24+ character) passwords.
* **Finding 5** – Remove this site from the public domain, it appears to be a dev testing environment.
* **Finding 6** – Ensure that **input** goes through validation and is then escaped or sanitized.
* **Finding 7** – Sanitize and validate inputs
* **Finding 8** – Properly secure the gitlab instance, the /explore endpoint was open to browse without using any credentials.
* **Finding 9** – **The safest way to prevent XXE is always to disable DTDs (External Entities) completely.**
* **Finding 10** – The primary defense is to avoid calling OS commands directly.
* **Finding 11** – To protect against verb tampering, user input should be properly validated and sanitized and use appropriate security measures
* **Finding 12** –

### Medium Term

* **Finding 1** – As an additional defense-in-depth measure, replace enumerable numeric identifiers with more complex, random identifiers. You can achieve this by adding a column with random strings in the database table and using those strings in the URLs instead of numeric primary keys. Another option is to use UUIDs or other long random values as primary keys. Avoid encrypting identifiers as it can be challenging to do so securely.
* **Finding 2** – The location where the files should be stored must be chosen based on security and business requirements. The following points are set by security priority, and are inclusive: 1. Store the files on a **different host**, which allows for complete segregation of duties between the application serving the user, and the host handling file uploads and their storage. 2. Store the files **outside the webroot**, where only administrative access is allowed. 3. Store the files **inside the webroot**, and set them in write permissions only. 4. If read access is required, setting proper controls is a must (e.g. internal IP, authorized user, etc.)
* **Finding 3** – The most effective solution to eliminate file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem/framework API. If this is not possible the application can maintain an allow list of files, that may be included by the page, and then use an identifier (for example the index number) to access to the selected file. Any request containing an invalid identifier has to be rejected, in this way there is no attack surface for malicious users to manipulate the path.
* **Finding 6** – When you need to safely display data exactly as a user types it in, output encoding is recommended. Variables should not be interpreted as code instead of text. If you’re not using a framework or need to cover gaps in the framework then you should use an output encoding library. Each variable used in the user interface should be passed through an output encoding function.
* **Finding 7** – Several protective measures are possible at the **Application** and **Network** layers. To apply the **defense in depth** principle, both layers will be hardened against such attacks. The first level of protection is input validation, as well, A regex can be used to ensure that data received is valid from a security point of view if the input data have a simple format (e.g. token, zip code, etc.). Otherwise, validation should be conducted using the libraries available from the string object because regex for complex formats are difficult to maintain and are highly error-prone.
* **Finding 9** – XML External Entity prevention depends on the technologies being used. A great resource for prevention can be found on the OWASP site. [OWASP XML Entity Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html)
* **Finding 10** – Built-in library functions are a very good alternative to OS Commands, as they cannot be manipulated to perform tasks other than those it is intended to do. Escape values added to OS commands. **Parameterization:** If available, use structured mechanisms that automatically enforce the separation between data and command. These mechanisms can help provide the relevant quoting and encoding. **Input validation:** The values for commands and the relevant arguments should be both validated. There are different degrees of validation for the actual command and its arguments:
* **Finding 11** – Use regex to check the data matches a specific pattern or by using a library that can validate data types. Access control is another important measure to protect against verb tampering. This involves ensuring that only authenticated users can access certain resources or perform certain actions. For example, a login page should only be accessible to users who have been authenticated, and a DELETE request should only be allowed for users with administrative privileges. Encryption is also an important measure to protect against verb tampering. By encrypting sensitive data, an attacker will not be able to view the data even if they are able to intercept the request. This can be done by using HTTPS, which encrypts the data sent between the client and the server. As well a WAF should be implemented.
* **Finding 12** – If this site is not in use anymore it should be removed. If it is still in use it should be updated to the current version of Drupal.

### Long Term

* Perform ongoing external network vulnerability assessments and password audits on all subdomains.
* Perform periodic updates on the technologies in use for the subdomains.
* Educate web developers on security hardening best practices compromise.
* Keep a close record of site in use and remove any that are not being used anymore.

## Remediation Summary -Internal Network

### Short Term

* **Finding 1** – Limit mounts and shares to users who absolutely need them.
* **Finding 2** – Set strong (24+ character) passwords on all SPN accounts.
* **Finding 3** – Patch the system.

### Medium Term

* **Finding 1** – Secure files and shares that have passwords in them to Administrator access only.
* **Finding 2** – Transition from SPNs to Group Managed Service Accounts (gMSA) wherever possible.
* **Finding 3** – Implement an update policy to make sure patches are applied.
* **Finding 4** – Enhance the domain password policy.
* **Finding 5** – Perform a network file share audit.
* **Finding 6** – Enhance network logging and monitoring.
* **Finding 6** – Implement an enterprise endpoint detection & response solution.

### Long Term

* Perform ongoing internal network vulnerability assessments and domain password audits.
* Perform periodic Active Directory security assessments.
* Educate systems and network administrators and developers on security hardening best practices compromise.
* Enhance network segmentation to isolate critical hosts and limit the effects of an internal compromise.

## Technical Findings Details

### Insecure Mount - High

|  |  |
| --- | --- |
| CWE | [CWE-522](https://cwe.mitre.org/data/definitions/522.html) |
| CVSS 3.1 Score | 9.5 |
| Description (Incl. Root Cause) | By having an unsecured mount on the system adversaries may be able to mount the file system and find critical information that can be used for lateral or vertical privilege escalation.  Mounts are file systems that can be mounted to the host system that may include sensitive information. |
| Security Impact | Adversaries may be able to mount these file share or drives and exfiltrate data as well as discover sensitive information including intellectual properties or sensitive data such and files and contain critical system information, such as passwords or technical information that can be used to further compromise the network or systems. |
| Affected Domain | * INLANEFREIGHT.LOCAL |
| Remediation | * Secure all file systems and mounts with a complex password and make sure the data is encrypted at rest. |
|  |  |

Finding Evidence:

Mounting the share to find a password in cleartext.



*Figure 71: Mounting the NFS share on DEV01*

Reading the file and finding the password for the Administrator on the DotNetNuke site.

A screen shot of a computer

Description automatically generated

*Figure 72: Finding credentials on web.config*

### Weak Kerberos Authentication (“Kerberoasting”) - High

|  |  |
| --- | --- |
| CWE | [CWE-522](https://cwe.mitre.org/data/definitions/522.html) |
| CVSS 3.1 Score | 9.5 |
| Description (Incl. Root Cause) | In an Active Directory (AD) environment, Service Principal Names (SPNs) are used to uniquely identify instances of a Windows service. Kerberos authentication requires that each SPN be associated with one service account (Active Directory user account). Any authenticated AD user can request one or more Kerberos Ticket-Granting Service (TGS) tickets from the domain controller for any SPN accounts. These tickets are encrypted with the associated AD account’s NTLM password hash. They can be brute forced offline using a password cracking tool such as Hashcat if a weak password is used along with the RC4 encryption algorithm. If AES encryption is in use, it will take more resources to “crack” a ticket to reveal the account’s clear-text password, but it is possible if weak passwords are in use. |
| Security Impact | A successful Kerberoasting attack along with cracked passwords could lead to lateral movement and privilege escalation in an AD environment. If a password is cracked for a Domain Administrator account or equivalent, an attacker could gain control over most, if not all, resources in the domain. |
| Affected Domain | * INLANEFREIGHT.LOCAL |
| Remediation | Where possible eliminate SPNs in the environment in favor of Group Managed Service Accounts (gMSA) which are not subject to this type of attack. If migration to gMSAs is not possible the following steps will help mitigate the risk of this attack:   * Enable AES Kerberos encryption instead of RC4 * Use strong 25+ character passwords for service accounts and rotate them periodically * Limit the privileges of service accounts and avoid creating SPNs tied to highly privileged accounts such as Domain Administrators |
| External References | <https://attack.mitre.org/techniques/T1558/003/> |

Finding Evidence:

Retrieving a listing of all SPN accounts in the INLANEFREIGHT.LOCAL domain using the powerview.ps1 from the PowerView toolkit.

A blue screen with white text

Description automatically generated

*Figure 73: Running powerview to check for Kerberoastable accounts on the host.*

Exporting the users and hashes to a CSV file so it can be moved offline to attempt password cracking.



*Figure 74: Exporting the SPN hashes to a csv file for cracking.*

Running hashcat on the file and finding the password for backupjob user/account.

A screen shot of a computer

Description automatically generated

*Figure 75: Cracking the hash on the SPN hashes.*

## Critical patches missing - High

|  |  |
| --- | --- |
| CWE | [CWE-522](https://cwe.mitre.org/data/definitions/522.html) |
| CVSS 3.1 Score | 9.5 |
| Description (Incl. Root Cause) | The DEV01 host was missing KB5005010. |
| Security Impact | Users having SeImpersonatePrivilege rights on the host can exploit this privilege and escalate their privilege to the nt authority\system account. |
| Affected Domain | * INLANEFREIGHT.LOCAL |
| Remediation | Patch the host with all critical updates from Microsoft |
| External References | <https://support.microsoft.com/en-us/topic/kb5005010-restricting-installation-of-new-printer-drivers-after-applying-the-july-6-2021-updates-31b91c02-05bc-4ada-a7ea-183b129578a7> |

Finding Evidence:

Running whoami /priv to see that the user has SeImpersonatePrivilege rights.

A screenshot of a computer

Description automatically generated

*Figure 76: Checking user privileges.*

Running the PrintSpoofer64.exe exploit to gain nt authority\system rights on the host.

A screen shot of a computer

Description automatically generated

*Figure 77: Exploiting PrintSpoofer64.exe to gain admin rights.*

## Weak Active Directory Passwords - High

|  |  |
| --- | --- |
| CWE | [CWE-521](https://cwe.mitre.org/data/definitions/521.html) |
| CVSS 3.1 Score | 9.5 |
| Description (Incl. Root Cause) | The tester found that users were using common, weak, passwords within the Active Directory domain and was able to uncover passwords for several users via a password spraying attack. Furthermore, an analysis of all domain passwords after achieving domain compromise showed more widespread weak password usage. |
| Security Impact | An attacker may be able to use this to guess passwords and gain a foothold within the internal environment. If external services are set up with Active Directory authentication (such as VPN, email, or remote application services) an attacker may be able to perform a targeted password spray to gain internal network access from an anonymous position on the internet. |
| Affected Domain | * INLANEFREIGHT.LOCAL   *See Appendix E – INLANEFREIGHT.LOCAL Domain Password Review for a detailed domain password analysis.* |
| Remediation | Review the password policy and enforce a 12-character minimum password. Consider implementing an enterprise password manager to encourage the use of strong, randomized, passwords. Implement a password filter to restrict the use of common words such as variations on the words “welcome” and “password”, seasons, months, and variations on the company name. |
| External References | <https://attack.mitre.org/mitigations/M1027/> |

Finding Evidence:

Running hashcat on the passwords found from running secrectsdump

A screenshot of a computer

Description automatically generated

*Figure 78: Running hashcat on the hashes found from running secrectsdump.*

## Insecure File Shares - Medium

|  |  |
| --- | --- |
| CWE | [CWE-284](https://cwe.mitre.org/data/definitions/284.html) |
| CVSS 3.1 Score | 6.2 |
| Description (Incl. Root Cause) | The tester uncovered multiple file shares where all Domain Users have read/write access. |
| Security Impact | An attacker who gains a foothold in this domain can use this access to search for files containing sensitive data such as credentials and potentially write malicious files to the file shares. |
| Affected Domain | * INLANEFREIGHT.LOCAL |
| Remediation | Review file share privileges to ensure that users are granted access in accordance with the principal of least privilege. |
| External References | <https://attack.mitre.org/techniques/T1135/> |

Finding Evidence:

Viewing file shares accessible to a standard Domain user with the [CrackMapExec](https://github.com/byt3bl33d3r/CrackMapExec/) tool.

A screenshot of a computer program

Description automatically generated

*Figure 79:* Viewing file shares accessible to a standard Domain user.

## Enhance Security Monitoring Capabilities - Info

|  |  |
| --- | --- |
| CWE | [CWE-693](https://cwe.mitre.org/data/definitions/693.html) |
| Description (Incl. Root Cause) | It appeared that Inlanefreight did not notice “noisy” activities during the course of testing. The tester was also not blocked when using standard open-source penetration testing tools. |
| Security Impact | If network and endpoint detection and response are inadequate, an attacker who can gain a foothold in the internal network may be able to move laterally, perform post-exploitation, and achieve persistence easily. |
| Remediation | Consider investing in a more advanced network monitoring solution, configuring logging on all hosts, and processing them for anomalies using a SIEM tool, and implementing endpoint detection on each server and workstation that is more difficult to bypass and tamper with. The organization should not rely on endpoint protection alone. When combined with a defense-in-depth security strategy, they can be an excellent tool for detecting an attacker who gains internal network access and is forced to perform “noisier” and riskier activities to the nature of the hardened environment. |
| External References | <https://attack.mitre.org/tactics/TA0005/> |

## Appendices

### Appendix A – Finding Severities

Each finding has been assigned a severity rating of high, medium, or low. The rating is based off of an assessment of the priority with which each finding should be viewed and the potential impact each has on the confidentiality, integrity, and availability of Inlanefreight’s data.

|  |  |
| --- | --- |
| Rating | Severity Rating Definition |
| **High** | Exploitation of the technical or procedural vulnerability will cause substantial harm. Significant political, financial, and/or legal damage is likely to result. The threat exposure is high, thereby increasing the likelihood of occurrence. Security controls are not effectively implemented to reduce the severity of impact if the vulnerability were exploited. |
| **Medium** | Exploitation of the technical or procedural vulnerability will significantly impact the confidentiality, integrity, and/or availability of the system, application, or data. Exploitation of the vulnerability may cause moderate financial loss or public embarrassment. The threat exposure is moderate-to-high, thereby increasing the likelihood of occurrence. Security controls are in place to contain the severity of impact if the vulnerability were exploited, such that further political, financial, or legal damage will not occur.  - OR -  The vulnerability is such that it would otherwise be considered High Risk, but the threat exposure is so limited that the likelihood of occurrence is minimal. |
| Low | Exploitation of the technical or procedural vulnerability will cause minimal impact to operations. The Confidentiality, Integrity and Availability (CIA) of sensitive information are not at risk of compromise. Exploitation of the vulnerability may cause slight financial loss or public embarrassment. The threat exposure is moderate-to-low. Security controls are in place to contain the severity of impact if the vulnerability were exploited, such that further political, financial, or legal damage will not occur.  - OR -  The vulnerability is such that it would otherwise be considered Medium Risk, but the threat exposure is so limited that the likelihood of occurrence is minimal. |

*Table 5: Severity Definitions*

## Appendix B – Exploited Hosts

|  |  |  |  |
| --- | --- | --- | --- |
| Host | Scope | Method | Notes |
| 172.168.8.3 (DC01) | Internal | DCSync | Domain compromise |
| 172.16.8.50 (MS01) | Internal | Credential Theft (ps1 file) | Domain lateral movement |
| 172.16.8.20 DEV01 | Internal | DotNetNuke credentials found on a share and SQL Console used to exploit the host | Alternate domain foothold |
| 172.16.8.12 (DMZ01) | External | Command Injection from the monitoring.inlanefreight.htb domain | Initial foothold |
| 172.16.9.25 (MGMT01) | Internal | SSH key found on the DC |  |

*Table 6: Exploitation Attempt Details*

## Appendix C – Compromised Users

|  |  |  |  |
| --- | --- | --- | --- |
| Username | Type | Method | Notes |
| MSSQLADM | Domain | SAM Dump on MS01 | Used in path for DCSync |
| BACKUPADM | Domain | Credential Theft (SQL Express Backup.ps1) on SMB share | User on MS01 |
| MMERTLE | Domain | Password Spray | Domain user |
| HPORTER | Domain | Credential Theft (SAM Dump) | Domain user for ForcePasswordChange over ssmalls user. |
| SSMALLS | Domain | PowerView (ForcePasswordChange) | Domain user with access to SMB shares that had credentials in them. |
| KDENUNEZ | Domain | Password Spray | Domain User |
| BACKUPJOB | Domain | Kerberoasting | Domain User |
| TTIMMONS | Domain | ForcePasswordChange | Used for DCSync |
| FRONTDESK | Domain | Password Spray | Domain User |

*Table 7: Accounts Compromised*

Table 6: User Accounts Compromised

## Appendix D – Changes/Host Cleanup

|  |  |  |
| --- | --- | --- |
| Host | Scope | Change/Cleanup needed |
| 172.16.8.50 (MS01) | Internal | ttimmons needs password reset |
| 172.16.9.25 (MMGT01) | Internal | exploit-1.c file in ssmalls home directory |

*Table 8: Assessment Artifacts*

Table 7: Assessment Artifacts

## Appendix E – INLANEFREIGHT.LOCAL Domain Password Review

### Password Statistics

|  |  |
| --- | --- |
| Metric | # |
| Total Password Hashes Obtained | 2917 |
| Total Passwords Cracked | 9 |
| % of Passwords Cracked | 0.3 % |
| Number of Domain Admins | 1 |
| Cracked Domain Admin Passwords | 1 |
| % of Domain Admin Passwords Cracked | 100 % |

Table 8: Password Cracking Statistics

### Most Commonly Used Passwords

|  |  |
| --- | --- |
| Metric | # |
| Welcome1 | 2 |

Table 9: Password Reuse Statistics

### Password Length Breakdown

|  |  |
| --- | --- |
| Length | # |
| 8 | 3 |
| 9 | 2 |
| 10 | 1 |
| 11 | 1 |
| 13 | 2 |
| **14** | 1 |
| 15 | 1 |
| 16 | 1 |
| 20 | 1 |

Table 10: Password Length Statistics